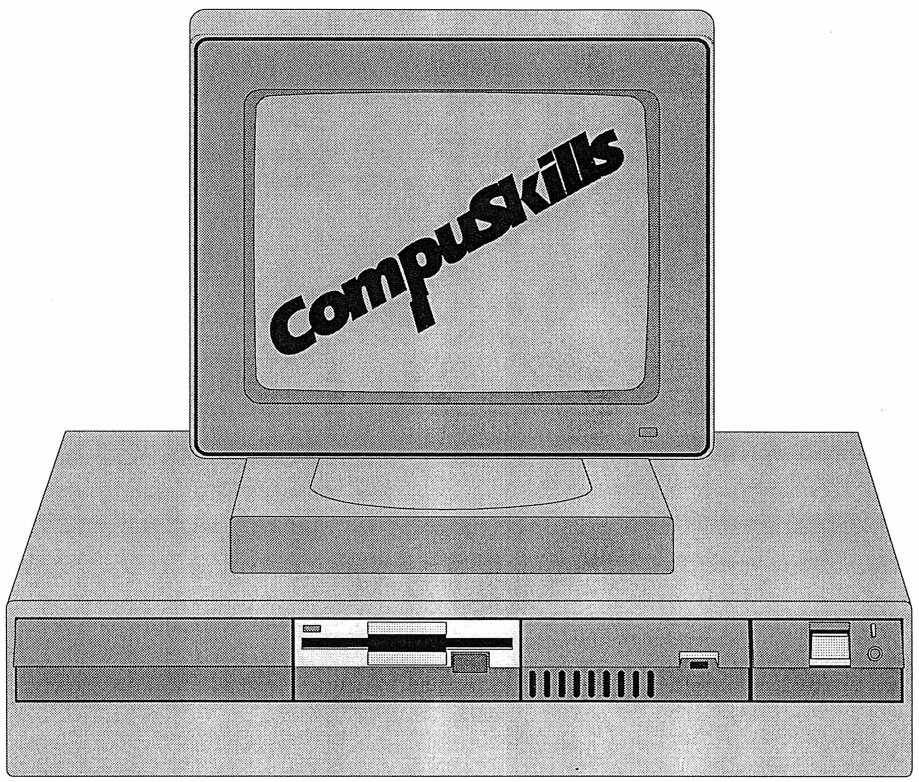
DOS

Level 2

CompuSkills
The Fast Track to Computer Literacy
An Affiliate of Colorado Free University
PO Box 6326, Denver CO 80206 (303) 329-6666

DOS

Level 2



CompuSkills
The Fast Track to Computer Literacy
An Affiliate of Colorado Free University
PO Box 6326, Denver CO 80206 (303) 329-6666

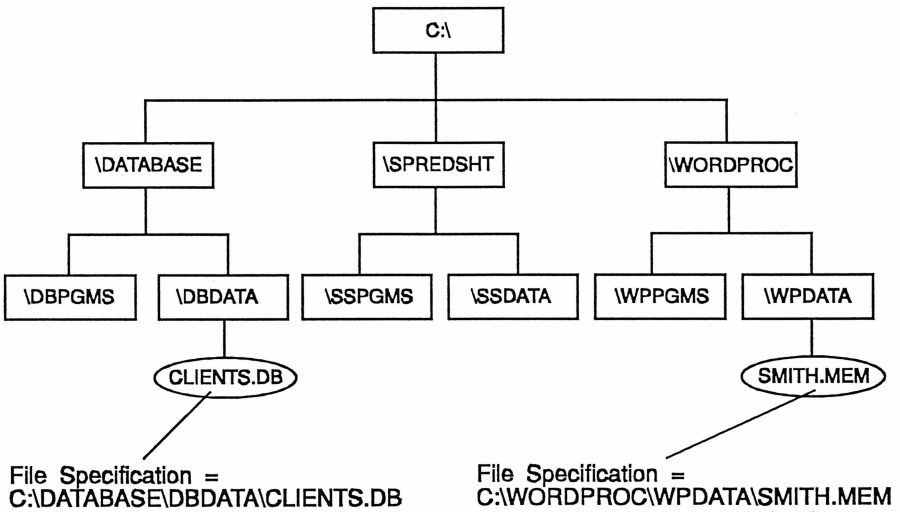
Contents

Section 1: Directories and DOS ---- Review

File Specifications

A file's name is only part of its identification. We must also tell DOS where to find the file. Therefore, the full file specification consists of:

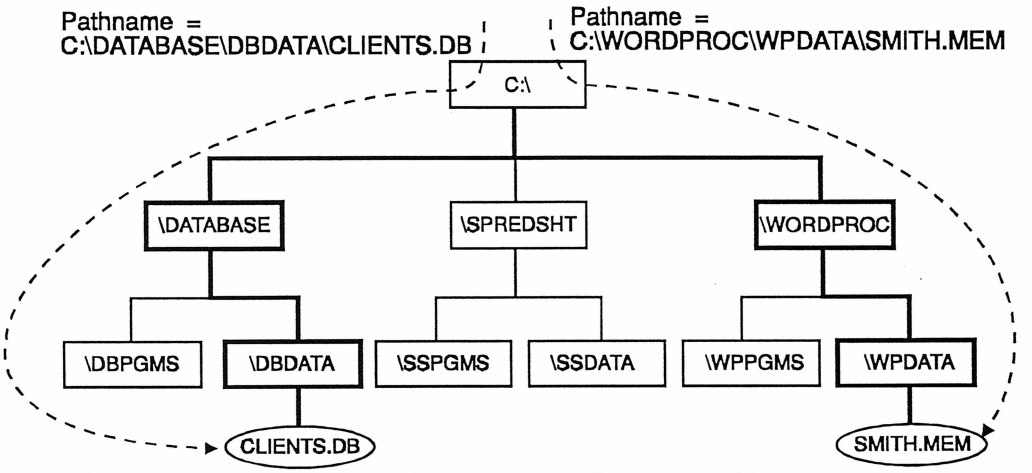
Drive letter + directory + filename.ext as in **C:\DATABASE\DBDATA\CLIENTS.DB**



File names must be unique within a directory, but not within the entire disk. Thus, we could have a file called SMITH.MEM in other directories.

Navigating the tree

In order to tell DOS how to get to a particular file, we must give it directions for navigating the tree. We call the route that DOS must take the **path,** or **pathname**. The pathname must include the names of all the directories that DOS must traverse to get to the file from the root directory.



To copy the client data base CLIENTS.DB to a floppy disk, we would use:

```
C:\>COPY  C:\DATABASE\DBDATA\CLIENT.DB  A:\
```

or, to copy a new customer data base from a floppy:

```
C:\>COPY  A:\CUSTOMER.DB  C:\DATABASE\DBDATA
```

Default (current) directory

Typing out long pathnames is very tedious and we might make mistakes. DOS gives us a short cut. We can tell DOS that a particular directory is the **CURRENT DIRECTORY**, or **DEFAULT DIRECTORY**.

If we make C:\DATABASE\DBDATA the current directory, then DOS assumes that every command we type refers to that directory and the files in it. This saves us retyping long directory names over and over.

We use the CHANGE DIRECTORY command to do this, shortened to CD:

```
C:\>CD \DATABASE\DBDATA     [note that we don't need to say C:]
```

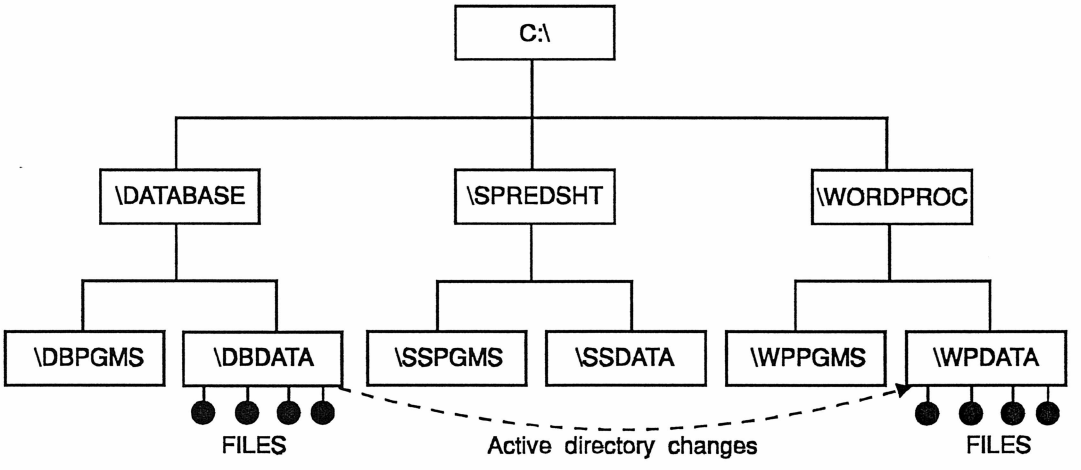
changes the default directory to C:\DATABASE\DBDATA.

So, if the current directory is \DATABASE\DBDATA, we can change the default again with:

```
C:\DATABASE\DBDATA>CD \WORDPROC\WPDATA

C:\WORDPROC\WPDATA>
```

and begin working on the files in \WPDATA



If we now copy files from A:\ without over-riding the default, DOS assumes that we mean the current directory for segment 3 of the command.

```
C:\WORDPROC\WPDATA>COPY  A:\*.*
```

causes DOS to copy all the files on A:\ to the directory \WORDPROC\WPDATA.

Prompt=

Knowing the current default directory is vital. We use the PROMPT command to set the DOS prompt to display the current directory:

```
PROMPT=$P$G
```

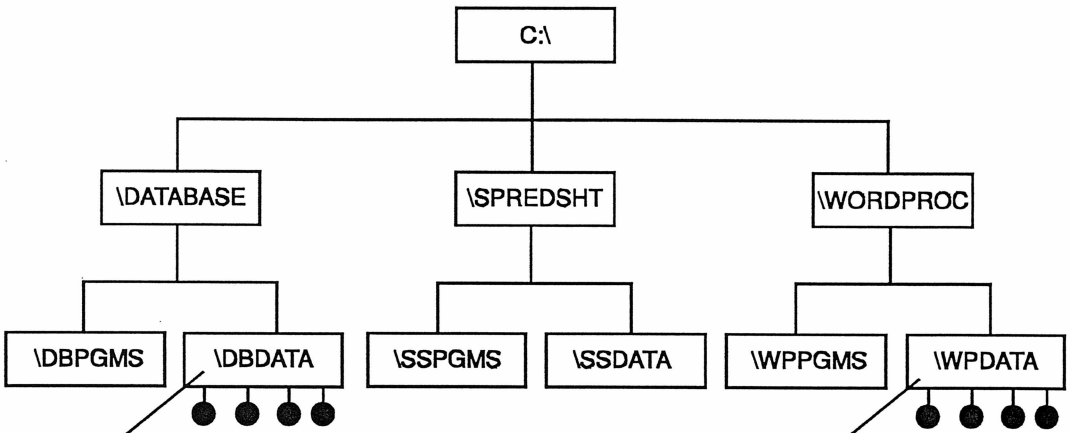
It's usual to keep this command in the AUTOEXEC.BAT file. Use **DOS Help** to see the options for the PROMPT command

Over-riding the default

We can still make commands that affect other directories simply by including the full path name in the command. This over-rides the default:

```
C:\WORDPROC\WPDATA>COPY  A:\*.*  \DATABASE\DBDATA

C:\WORDPROC\WPDATA>COPY  \DATABASE\DBDATA\CLIENT.DB  A:\
```



With a DOS prompt of:

C:\WORDPROC\WPDATA>

we must over-ride the default to specify files in this directory

With a DOS prompt of:

C:\WORDPROC\WPDATA>

we DO NOT over-ride the default to specify files in this directory

When do we change the default?

The benefit of setting a default is that it saves us typing time and increases accuracy when we are doing a lot work in one directory. If our current directory is \DATABASE\DBDATA and we want to see a list of all the files in WORDPROC\WPDATA, then we could **over-ride** the default for that one command:

```
C:\DATABASE\DBDATA>DIR  \WORDPROC\WPDATA
```

and DOS would list the files without changing the current directory.

But if we've finished in one directory and intend to do a lot of work in another, like \WORDPROC\WPDATA, then we would change the default:

```
C:\DATABASE\DBDATA>CD \WORDPROC\WPDATA
```

Notice that if one or both directories are on the active drive, we don't need to include the drive letter. But if we need to refer to another drive, we must include its drive letter:

```
C:\WORDPROC\WPDATA>COPY  A:\*.*  \DATABASE\DBDATA
```

Directories and Program Files

The idea of the current directory also applies to program files. If we issue a command to tell DOS to execute a program file, we must tell DOS where to find it.

Suppose we have a program file called DRAW.EXE in a directory called \GEMDRAW. We cannot just enter:

```
C:\>DRAW
Bad command or file name
```

because DOS cannot find it in the root directory, C:\

e must enter:

```
C:\>CD \GEMDRAW
C:\GEMDRAW>DRAW
```

Path Statement

It's very inconvenient to have to change the directory in order to start a program, so we prepare a list of directory pathnames where DOS should look and store it in the **autoexec.bat** file. This is called a PATH statement:

```
PATH = C:\DBASE;C:\DOS;C:\GEMDRAW;C:\LOTUS;C:\WP;C:\VENTURA
```

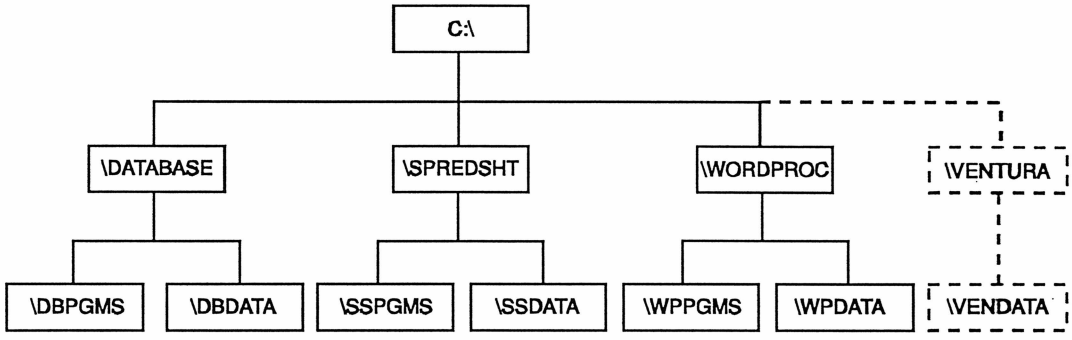
It's not a very pretty list: no spaces and each pathname separated by a semicolon. But, it tells DOS to look in each of the listed directories for the execute file.

When we buy a new program, the installation program may create a new directory and update the PATH statement for us. But sometimes, we may need create the new directory for it, so remember to include the directory name, like C:\NEWDIR\PROGRAM, in the PATH statement in the **autoexec.bat** file. (We'll see how later.)

Making directories

To make a new directory, we use the MAKE DIRECTORY command, or MD for short:

```
C:\>MD \VENTURA
```



and we create subdirectories in the same way:

```
C:\>MD \VENTURA\VENDATA
```

If we make the parent-to-be the current directory, we do not need to include the parent's name:

```
C:\VENTURA>MD VENDATA
```

but we must not use a backslash in the new subdirectory's name, otherwise DOS will think that we want to create it under the root. For example:

```
C:\VENTURA>MD \VENDATA
```

will create a new directory immediately under the root directory.

Removing directories

If we no longer need a directory, we can delete it. But first, we must erase all its files. Then, we must go up at least one level in the structure (we can't remove the current directory). We use the REMOVE DIRECTORY command, or RD for short:

```
C:\VENTURA>RD VENDATA
```

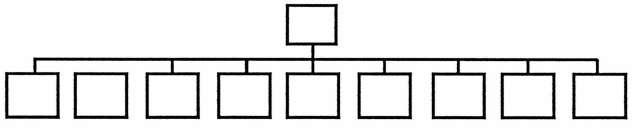
Managing The Hard Disk

Setting up a new computer system, especially the hard disk, is fairly easy, but very important. Here are some tips:

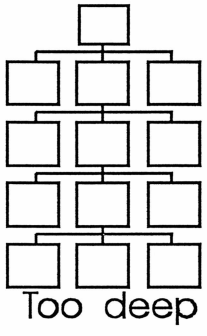
1. Plan ahead.

- **What kinds of data files will we store on our hard disk?**
  Will it be mostly word processing documents, spreadsheets, data bases, graphics, or a mixture? It's a good idea to keep the same kind of files together to make it easier to find them later, and to back up our directories to floppy disk.

- **Keep a balance between the width and depth of the directory tree.**
  If it's too broad (more than 10 - 15 subdirectories immediately under the root directory), we won't be able to see them all on the screen when we use the DIR command, and the tree structure we see in the DOS shell will be long and stringy. If our tree is too deep (more than three levels), we'll have to type long path names.



Directory structure too broad



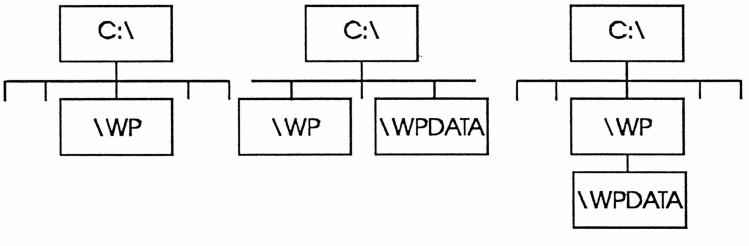
Too deep

- **Keep data and program files separate.**
  When we install a new application program, the install procedure usually makes a directory and loads the program files to it. Unless we tell the application otherwise, it will save our data files to this same directory. Not a good idea! Mixing program and data files is bad news because:
  - It's more difficult to scan the list of files looking for a particular data file
  - Back up utilities like Fastback or Mace work best on whole directories, but we don't want to back up our program files
  - If we want to upgrade to a newer version of the software later, we can't just clear the old version's directory and start over because it also contains our data files.

Separate data and programs

If we have our WordPerfect programs in **C:\WP**, then we have two options for a data directory:

- A directory immediately under the root called **C:\WPDATA**
- A subdirectory under C:\WP called **C:\WP\DATA**.



2. Create the directories.

The dealer will probably have set up our hard disk with the root directory and a directory called \DOS to hold the DOS files. Our job is to create the other directories that we need.

- First we load the software applications on the disk from the floppy disks. The installation program will usually create the directory(s) needed by the program so that it can copy the application programs to it.

- Second, starting from the root, we make the data directories we need using the M(ake) D(irectory) command. (Alternatively, we can do this as part of the application set up.)

3. Keep the directories clean.

1. Treat the root directory like a master index to the disk. Do not clutter it up with data files. Apart from a few DOS files, the root directory should be clear of files. (One person we know complained that the computer was slow to retrieve files. No wonder! The root directory had over 2000 files in it!)

2. Periodically, use the DIR command, or scan the directories using the DOS shell, Windows' File Manager, or a utility like XTREE looking for "dead" files. You'll be surprised how much garbage collects.

3. Defragment the hard disk regularly, say, monthly to counteract DOS' tendancy to break files up so as to put them into available empty space. See DOS Help for details of the DEFRAG command.

Note: DEFRAG performs the same function as Norton's SPEEDISK, but if you are using a disk compression utility like DOUBLESPACE or STACKER, you must use the defragmenting utility provided by the manufacturer otherwise SERIOUS DATA LOSS WILL OCCUR.

Section 2: Controlling DOS

Although controlling DOS may seem like a contradiction in terms, DOS provides many tools for configuring our system to our particular needs and liking. We put the tools in two files that DOS reads and executes when it boots the computer up. They are:

- **CONFIG.SYS**
- **AUTOEXEC.BAT**

CONFIG.SYS

For DOS to work with our computer, it must know what kinds of devices (like a mouse) we have hooked up. We include this information in a **SYS**tem **CONFIG**uration file. This file must live in the root directory, and typically might contain:

```
DEVICE=C:\DOS\HIMEM.SYS

DEVICE=C:\DOS\EMM386.EXE NOEMS

BUFFERS=10,0

FILES=70

DOS=HIGH,UMB

LASTDRIVE=Z

FCBS=16,8

DEVICE=C:\DOS\SETVER.EXE

DEVICEHIGH /L:3,2496 =C:\KW559AG.SYS

SHELL=C:\DOS\COMMAND.COM C:\DOS\  /p

DEVICEHIGH /L:2,39488 =C:\DOS\DBLSPACE.SYS /MOVE
```

These cryptic little messages tell DOS what kind of system this is. For example:

- **DEVICE=HIMEM.SYS** tells DOS that we have additional memory (> 640KB) and need this extended memory manager to handle memory above 1MB. To DOS, the manager is just another device.
- **DEVICE=EMM386.EXE NOEMS** tells DOS that we also want it to load an upper memory manager to handle the memory between 640K and 1MB. **NOEMS** instructs DOS on how we want the memory made available.
- **BUFFERS=10,0** tells DOS how many blocks of memory to set aside for data. This is temporary storage that DOS uses while reading and writing to disk drives, and being able to store data in buffers allows DOS to run faster.
- **FILES=70** tells DOS the maximum number of files we expect to have open at any one time. DOS maintains a table of up to 255 open files, but the greater the number, the more RAM is used by the table. If we omit the FILES command, DOS allows the minimum of 8 files, and if we need more than that, the program will not run. If we're using large applications under Windows, 60 - 70 is a safe range of numbers to use.
- **DOS=HIGH, UMB** tells DOS to load part of itself in high and upper memory (i.e. memory above 640KB) in order to free up some conventional memory.

Other **DEVICE=** commands tell DOS to load other drivers—special programs that manage hardware and provide system facilities. Each CONFIG.SYS file is slightly different depending on exactly what hardware you have.

When we change our computer configuration, like adding a new mouse or adding more memory, we must record the new configuration in the CONFIG.SYS. Fortunately, many devices use installation programs that do this automatically. If they don't, the installation instructions tell us exactly what to do.

AUTOEXEC.BAT

The **autoexec.bat** file **AUTO**matically **EXEC**utes a **BAT**ch of commands whenever we boot the computer. (With other batch files, we must invoke them by typing their filename.) The autoexec.bat file must be in the root directory, and a typical autoexec.bat file might contain the following commands:

```
C:\WINDOWS\SMARTDRV.EXE

PROMPT $p$g

PATH=C:\; C:\DOS;C:\WINDOWS;C:\SCAN;C:\WINWORD;C:\TRAKKER;C:\CALERA

SET TEMP=C:\WINDOWS\TEMP

C:\DOS\doskey

C:\WINDOWS\MSCDEX.EXE /S /D:MSCD000 /M:10

C:\DOS\SHARE.EXE /L:500 /F:5100

c:\mouse\mouse.com
```

This sequence of commands customizes the way DOS works to your preference. The commands mean:

- **C:\WINDOWS\SMARTDRV.EXE** runs a program that creates a Smartdrive disk cache
- **PROMPT PG:** set the DOS prompt to be **C:\current directory>**
- **PATH:** We start a program by typing the first part of the filename of the main execute file, such as "WP" for WP.EXE. DOS then searches the current directory for the file. So we create a PATH command containing a list of all the directories that contain our programs. From then on, whenever we give DOS a command, it looks in the current directory and then each directory on the PATH list for the programs to execute. Separate the directory entries by semicolons, and the last entry has no semicolon. When you buy new software, the program's installation program usually adds its own directory to the list. If not, you should add the directory using a DOS text editor.]
- **SET TEMP=C:\WINDOWS\TEMP** gives Windows the location of the directory for its temporary files.

Other commands, such as **C:\MOUSE\MOUSE.COM**, invoke software such as the mouse driver. (The installation process for the mouse put this command in the autoexec.bat file.)

Multiple Configurations

DOS 6.0 allows us to define more than one configuration in CONFIG.SYS. This can be useful if you share a computer will several people or if you want to be able to start your own computer with a choice of configurations such as networked and non-networked. If your CONFIG.SYS file defines multiple configurations, DOS displays a menu that enables you to choose the configuration you want to use each time you use your computer.

1. **Define a startup menu in your CONFIG.SYS file.**
   Each time your computer starts, the startup menu appears and lists the available configurations; you choose the configuration you want from the menu.

2. **Create a configuration block in your CONFIG.SYS for each configuration you want**
   A configuration block begins with a block header, i.e. a name surrounded by square brackets. Each block contains the CONFIG.SYS commands that you want DOS to carry out when that configuration is selected from the startup menu.

3. **Control AUTOEXEC.BAT with conditions**
   DOS can also execute different AUTOEXEC.BAT commands for each startup configuration. Use batch commands such as **if** and **goto** to create conditional branches in your AUTOEXEC.BAT file.

Command Bypass

DOS 6 provides the ability to bypass startup commands when you tum on your computer. During booting, you can use this feature to choose which CONFIG.SYS commands DOS should carry out and whether DOS should run your AUTOEXEC.BAT. By controlling the commands your computer uses to start up, you can pinpoint problems more quickly and easily.

To start your computer without naming the commands in your CONFIG.SYS and AUTOEXEC.BAT you can bypass startup commands in the following ways:

- **Full Bypass**
  You can bypass all the commands in both your CONFIG.SYS and AUTOEXEC.BAT files
- **Specific Bypass**
  You can prevent DOS from carrying out specific CONFIG.SYS commands and specify whether or not DOS should run the AUTOEXEC.BAT file.
- **Conditional Bypass**
  You can have DOS prompt you to confirm a particular CONFIG.SYS command every time your computer starts. For example, to have DOS prompt you to confirm that command **device=c:\dos\ramdrive.sys** be executed each time your computer starts, change the command to **device?=c:\ dos\ ramdrive.sys.**

Note: For security purposes, you can prevent yourself or others from bypassing startup commands when your computer starts. To do this, add the **switches /n** command to your CONFIG.SYS file. For more information, type **help switches** at the command prompt.

Complete Bypass

During startup, when DOS displays the message: "Starting MS-DOS..." press F5 or hold down the Shift key. DOS displays: "MS-DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files."

The machine will start with a basic configuration instead of your usual configuration. Therefore, some parts of your system might not work as they usually do. For example:

- DOS might be unable to find COMMAND.COM and will display "Bad or missing command interpreter'. DOS will prompt you to specify the path to COMMAND.COM. Type the full path to the file, for example, C:\DOS\COMMAND.COM
- DOS will not load installable device drivers. As a result, any device that requires an installable device driver will not work. For example, your mouse will not work. Also, programs that require expanded or extended memory will not run because DOS cannot load expanded- or extended-memory managers.
- DOS will set environment variables to their default values. The command prompt might not appear the way it usually does. The search path will be set to the directory that contains DOS files (usually C:\DOS).

CONFIG.SYS Command Confirmation

If you are having problems that you suspect are related to a specific CONFIG.SYS command, get DOS to prompt you to confirm each command when your computer starts.

Restart or restart your computer. When DOS displays "Starting MS-DOS..." hit F8. DOS displays: "DOS will prompt you to confirm each CONFIG.SYS command." One at a time, DOS displays each command in your CONFIG.SYS followed by a prompt. For example:

DOSHIGH [Y,N]?

Press Y for Yes and N to bypass that command. To bypass all remaining startup commands, press F5. When DOS finishes processing the CONFIG.SYS file, it asks:

"Process AUTOEXEC.BAT [Y,N]?

To carry out all the commands in your AUTOEXEC.BAT, press Y for Yes. To bypass your AUTOEXEC.BAT completely, press N for No.

Section 3: Advanced DOS Commands

There are lots more DOS commands! In this section, we'll look at: XCOPY, BACKUP & RESTORE, ATTRIB, DOSKEY,MOVE, DELTREE, and how to create batch files.

XCOPY

The **COPY** command copies single files or groups of files if we use wildcards in filename, but COPY is limited to one directory only (either the named directory or the default directory).

XCOPY is different because it can also copy files in subdirectories (except hidden and system files). With this command, you can copy all the files in a directory, including the files in the subdirectories of that directory.

Syntax:

```
XCOPY source [destination] [/A|/M] [/D:date] [/P] [/S [/E]] [/V] [/W]
```

source:

Specifies the location and names of the files you want to copy. Source must include either a drive or a path.

destination:

Specifies the destination of the files you want to copy. Destination can include a drive letter and colon, a directory name, a filename, or a combination. If omitted, DOS will default to the current drive and directory.

Switches:

/A copies only source files that have their archive file attributes set. This switch does not modify the archive file attribute of the source file.

/M copies source files that have their archive file attributes set. Unlike the /A switch, the /M switch turns off archive file attributes in the files specified in source.

/D:date copies only source files modified on or after the specified date. Note that the format of date depends on the COUNTRY setting you are using.

/P prompts you to confirm whether you want to create each destination file.

/S copies directories and subdirectories, unless they are empty. If you omit this switch, XCOPY works within a single directory.

/E copies any subdirectories, even if they are empty. You must use the /S switch with this switch.

/V verifies each file as it is written to the destination file to make sure that the destination files are identical to the source files.

/W displays the following message and waits for your response before starting to copy files: "Press any key to begin copying file(s)"

XCOPY Examples

Suppose we have three subdirectories under C:\WP_DATA and we want to make a copy on a floppy disk for safety. We would use:

```
C:\>XCOPY \WPDATA  A:\ /S
```

/S is a **switch** that tells DOS to copy files in subdirectories as well as C:\WP_DATA.

/D tells XCOPY to copy only files written after a certain date:

```
C:\>XCOPY \WPDATA A:\  /D:06/06/91 /S
```

DOS Backup & Restore

A backup is a copy of data on the hard disk that we make for safety. Hard disks can self-destruct for many reasons—the head may touch the spinning surface and scratch it; the file tables can be corrupted; the disk controller electronics can break down, and so on.

If we back up our hard disk weekly and suffer a disk crash, we can replace the hard disk for a few hundred dollars, reload our application programs from the original floppy disks, and then **restore** our data files from our backup copies. But if we haven't made back up copies of our data files, we may have lost all our data.

Backing up is like buying car insurance. We do it but hope we'll never need it. If we need it but haven't done it, we're in big trouble.

DOS Setup installs BACKUP and RESTORE commands in DOS and Windows versions. BACKUP files are specially formatted and compressed, and we can't do anything with them except use the RESTORE command on them.

BACKUP

Backup offers three types of backup:

- **Full backup** backs up all the files that you select before starting the backup. A full backup can include all the files on your hard disk, but more commonly includes all files of one type, all files on a particular drive, or all files in one or more directories. Ensures that you have a copy of every file that you might need to restore.
  Good: Can restore any file easily.
  Bad: Backing up a large amount of data can be time-consuming and require more space on backup media than other methods.

- **Incremental backup** backs up only files that have changed since your last full or incremental backup. Records changes since the last full or incremental backup.
  Good: Because an incremental backup backs up only files that have changed, it quickly and fully protects you against data loss. Preserves multiple versions of a file. Is fastest if you work with many different files. Requires less space on your backup media.
  Bad: You need to keep each incremental backup (between full backups) because the backups build on each other.

- **Differential backup** backs up only the files that have changed since your last full backup.
  Good: You need to keep only the last full and the last differential backup sets to restore files.
  Bad: A differential backup may take a little longer to complete than an incremental backup. Can become large if you work with many different files. Cannot retrieve previous versions of a file.

Backing up

Enter the command:

```
C:\>MSBACKUP
```

DOS replies with a dialog box in which we specify BACKUP or RESTORE. If we choose BACKUP, another dialog box appears in which we select the source files to be backed up, the target floppy disk drive, and whether we want full, incremental, or differential backup.

DOS lists the files being backed up. If the files overflow the first disk, DOS asks us to put another disk into the floppy drive, and so on until the entire directory has been backed up.

Restore

Restoring files is equally as easy.

enter the command:

```
C:\>MSBACKUP
```

DOS replies with a dialog box in which we specify BACKUP or RESTORE. If we specify RESTORE, another dialog box appears in which we select the source floppy disk drive, the source files to be restored, and the target directory.

ATTRIB and File Attributes

Examine the backup files you made on your floppy disk and try to delete them. DOS denies access to them because they are important files and should not be inadvertantly deleted. When DOS creates these files, it gives them a **read only attribute**.

Files can have four attributes:

- **A = Archive.** If set, this indicates that the file has been changed since the last time the directory was backed up. DOS BACKUP refers to this attribute to see if it should back up the file on differential and incremental backups.
- **R = Read Only.** The file can be read but not written to. (DOS Backup sets this attribute automatically.)
- **H = Hidden File.** This prevents the file from being listed when we use the DIR command, and prevents it from being deleted.
- **S = System File.** This indicates that the file is a DOS system file like IO.SYS and should be kept in a particular place on the disk (called the boot sector). Also, if we make a **boot disk** using the command **sys a:**, DOS copies all files with an S attribute to the floppy disk.

ATTRIB Command

We can look at a file's attributes using the ATTRIB command:

```
C:\>ATTRIB C:\*.*

C:\>ATTRIB A:\*.SYS
```

will produce a list of files in the C:\ directory and their attributes, or all files on the A: drive with an extension of .SYS.

We can change a file's attributes using the ATTRIB command:

```
C:\>ATTRIB +R filename.ext
```

makes the file read only. To make it read/write once more, use:

```
C:\>ATTRIB -R filename.ext
```

removes the read only attribute.

We can change all the attributes in one command:

```
C:\ATTRIB +R +A +S +H filename.ext
```

To delete the backup files we made:

```
C:\>ATTTRIB -R  A:\*.*
C:\>DEL A:\*.*
```

MOVE Command

We use the MOVE command (DOS6 and later) to move one or more files and to rename a directory:

```
MOVE [drive:][path]filename1 destination
```

where **[drive:][path]filename1** specifies the location and name of the file or files you want to move, and **destination** specifies the new location of the file. Destination can consist of a drive letter and colon, a directory name, or a combination. If you are moving only one file, you can also include a filename if you want to rename the file when you move it.

For example, suppose C:\LETTERS is a directory. To move the files ED.TXT and SIGRID.TXT from the current directory to the LETTERS directory on drive C, type:

```
MOVE ED.TXT,SIGRID.TXT C:\LETTERS
```

To move the BILL.TXT file from the current directory to the LETTERS directory on drive C and rename it ANN.TXT, type:

```
MOVE BILL.TXT C:\LETTERS\ANN.TXT
```

To rename a directory:

```
MOVE [DRIVE:][PATH]DIRNAME1 DIRNAME2
```

where **[drive:][path]dirname1** specifies the directory you want to rename, and **dirname2** specifies the new name of the directory.

For example, to rename the THISYEAR directory on drive C to LASTYEAR, type:

```
MOVE C:\THISYEAR C:\LASTYEAR
```

DELTREE Command

DELTREE deletes a directory and all the files and subdirectories that are in it.

```
DELTREE [drive:]path
```

where **drive:path**specifies the name of the directory you want to delete. The command will delete all the files contained in the directory youspecify, as well as all subdirectories and files in the subdirectories subordinate to this directory. You can specify more than one directory.

For example, to delete the TEMP directory on drive C, including all files and subdirectories of the TEMP directory, type:

```
DELTREE C:\TEMP
```

DOS Batch Files

DOS stores the files it uses in a primitive format called ASCII or text file format, that has no formatting or special codes used by word processors. Word processors can also generate text files in addition to the highly formatted files they normally produce.

For example, in order to invoke WordPerfect, we normally use the CD command to make the \WP51 directory the default, and then we type WP to start the program.* Rather than typing this every time we want to use WordPerfect, we could create a small batch file called, say, **WORDPERF.BAT** that contains four lines:

```
CD \WP51
WP
CD \MENU
MENU
```

Now when we type WORDPPERF, we're actually telling DOS to execute the batch file called WORDPERF.BAT. DOS performs our little batch of instructions for us, changing the default and starting WordPerfect. While WordPerfect is executing, the batch file is suspended, but when we quit from WordPerfect, DOS resumes executing the batch file and changes the default to the \MENU directory and starts the MENU program.

Creating batch files

We can build a batch file of any sequences of DOS commands that we can type from a keyboard. Batch file names must have the .BAT extension.

COPY CON

We could create the WORDPERF.BAT file by telling DOS to copy whatever we type next on the keyboard (or CON for console) into a file called WORDPERF.BAT:

```
C:\>COPY CON WORDPERF.BAT
```

We type the commands exactly as we want them to appear, and we use **Ctrl + Z** to tell DOS that we've finished.

---

* Note that in real life, we would include the directory C:\WP51 in the autoexec.bat PATH statement.

EDIT

Occasionally we need to edit batch files like our WORDPERF.BAT, and the **autoexec.bat** and **config.sys** files. DOS provides a crude text editor program called EDIT.

To edit an existing file:

```
C:\>EDIT \directory\filename
```

Suppose we wanted to add another command (ECHO OFF) as line 1 of WORDPERF.BAT:

```
C:\>EDIT WORDPERF.BAT
```

We see the file displayed:

```
CD \WP51
WP
CD \MENU
MENU
```

and type the new line 1:

```
ECHO OFF
```

Now, if we press Alt, F, S to save our updated file. Now enter

```
C:\>TYPE WORDPERF.BAT

1 ECHO OFF
2 CD \WP51
3 WP
4 CD \MENU
5 MENU
```

Creating Batch Files

DOS executes four kinds of commands. If you enter the name of a file that ends with **.COM, .EXE,** or **.BAT**, DOS will try to run that program (e.g., **WP.EXE** or **MP.COM**) or execute the commands contained in that batch file (e.g., **DIRSORT.BAT**). DOS also recognizes its own set of internal and external commands.

A batch file is a file that contains one or more DOS commands. Batch files (batch programs) may also include a fourth kind of command understood by DOS: batch commands.

Batch commands direct or control the activity being carried out in a batch program:

Call starts another batch program without relinquishing control to that file.

Echo displays the following characters as text on the screen.

For runs a specified command for each file in a set of files.

If introduces a conditional command.

Goto sends the command interpreter to another line in the file.

Pause halts command execution until the user presses another key.

Rem or **remark** indicates that the remaining text on the line is not a command and can be ignored.

Shift changes the position of replaceable parameters in a batch program.

Options for the IF Command

IF VARIABLE == Constant tests to see if the variable on the left of the double equal signs is the same as the constant listed on the right side. If it is true and they are equal, the second part of the phrase is executed. If they are not equal, the command is ignored and processing continues to the next step.

IF EXIST Variable/Condition tests to see if the variable or condition exists, and then executes the command.

IF NOT EXIST Variable/Condition tests to see if the variable or condition does not exist, and then executes the command.

IF ERRORLEVEL # tests to see if the number of errors in a program has reached a specific amount, and then executes the command.

Examples of the IF Command

IF %1 == JAN88.WK1 COPY %1 A: If the substituted variable equals **JAN88.WK1**, then copy the file to drive A.

IF %1. == . GOTO :ENDING (tests for a blank entry). If the substituted variable is a blank, the left side of the equation will result in only a period (.); this condition will then equal the right side of the equation. Since the IF statement is now true, and the left and the right sides of the equation are equal, continue processing the batch file at the line named :ENDING.

IF NOT EXIST %1 ECHO You typed in a file that does not exist. If the variable you specified does not exist, display the message in the **ECHO** command.

IF NOT EXIST %1\NUL GOTO :ERROR If the directory you specified does not exist, continue processing the batch file at the line named :ERROR. (**DIRECTORY\NUL** is used to test for the existence of a directory).

IF ERRORLEVEL 1 GOTO :STOP Upon encountering the first error, go to the line named :STOP.

Procedures for Creating a Batch File

You can write a batch program using any text editor or word processor that saves files as ASCII text. You can use the DOS 5.0 text editor to produce a simple ASCII file.

1 .In your batch program, type the DOS commands to be included in a batch file, each on a separate line.

2. Save the file with the filename extension **.BAT.**

3. Run the batch file by typing the file name at the command prompt.

Practice 1 - Create a Batch File

We are going to create a batch file that displays a directory in two ways: a sorted list of just the subdirectories, and a sorted list of just the files in a directory.

Task/Comments	`Keystrokes`
If Doskey is not installed, install it. This will save you some typing time later.	`doskey`
At the command prompt, start the DOS text editor Edit to edit a new file called SORTDIR.BAT.	`edit SORTDIR.BAT`
Enter the command to have the batch file clear the screen.	`cls`
Enter the two directory commands to list and sort first subdirectories and then files of the directory \WP51. Have the commands pause the directory listings also.	`dir \WP51 /a:d/o/p` `dir \WP51 /a:-d/o/p`
Exit and save the file.	`File...Exit` `[Enter] to Save`
Try the batch file.	`sortdir`

Practice 2 - Pausing a Batch File

We are going to insert a pause in between the two commands so that you can read the screen.

Task/Comments	Keystrokes
Edit SORTDIR.BAT.	`edit SORTDIR.BAT` or if Doskey is installed hit the [Up] arrow until command appears at prompt line

Insert a pause command between the two directory commands.	Change your file to read:
	`cls`
	`dir \WP51 /a:d/o/p`
	`pause`
	`dir \WP51 /a:-d/o/p`

Exit and save the file.	
Run the batch file.	`sortdir`

Practice 3 - Adding Messages to Batch Files

You can add messages to a batch file to tell the operator what is happening or what to do, such as turning on the printer. Insert a pause after the message to give the operator time to read the message and respond.

Task/Comments

Keystrokes

Edit SORTDIR.BAT.

Use the echo command to add two messages that tell the user what's happening. Follow the first message with **pause** to give time to read the message

Change your file to read:

```
cls
echo  Subdirectories in \WP51
pause
dir \WP51 /a:d/o/p
echo  Files in \WP51
pause
dir \WP51 /a:-d/o/p
```

Exit and save the file.

Run the batch file.
(Note that the messages display twice because we first see the echo command displayed and then the command is carried out. We are going to use the **echo off** command so that we display only the results of the command.

Edit SORTDIR.BAT.

Push the text down one line and insert an echo off command at the top of the batch file.

Your batch file should read:

```
echo off
cls
echo  Subdirectories in \WP51
pause
dir \WP51 /a:d/o/p
echo  Files in \WP51
pause
dir \WP51 /a:-d/o/p
```

Save, exit, and try the batch file.

```
sortdir
```

Practice 4 - Adding Remarks and Blank Lines

When you run the batch file at this point, the display of second directory executes on the next line after the first directory command finishes. We are going to use the echo command with a period (.) to create blank lines on the screen to separate the execution of these commands. Rem commands are remarks added to a batch file that do not display or execute. They can be used to make comments about the commands in a batch file. We are going to add some remarks to our batch file.

Another use of the rem command is to force the batch file to ignore a command. This is useful if you don't want a command to execute but you don't want to delete it. Simply add the word rem and a space at the beginning of the command.

Task/Comments	`Keystrokes`

Edit SORTDIR.BAT.
Add **rem** in front of each dir command to describe what the /a switch does.
	Your program should read:
	`echo off`
	`cls`
	`echo  Subdirectories in \WP51`
	`pause`
	`rem The /a:d switch only lists directories`
	`dir \WP51 /a:d/o/p`
	`echo  Files in \WP51`
	`pause`
	`rem The /a:-d switch only lists files`
	`dir \WP51 /a:-d/o/p`

Use **echo** to create three blank lines between the execution of the first command and the second.	Your program should read:
	`echo off`
	`cls`
	`echo  Subdirectories in \WP51`
	`pause`
	`rem The /a:d switch only lists directories dir \WP51 /a:d/o/p`
	
	`echo.`
	`echo.`
	`echo.`
	`echo  Files in \WP51`
	`pause`
	`rem The /a:-d switch only lists files`
	`dir \WP51 /a:-d/o/p`

Exit and save the file.	
Run the batch file.	`sortdir`

Practice 5 - Using a Replaceable Parameter

Our batch file is useful, but it would be more useful if it wasn't limited to just the \WP51 directory. We are going to substitute \WP51 with a replaceable parameter, so that we can specify at run time which directory should be listed.

Task/Comments	Keystrokes
Edit SORTDIR.BAT.	

Replace each occurrence of \WP51 with %1, the symbol for the first replaceable parameter.	Your program should read:
	`echo off`
	`cls`
	`echo  Subdirectories in %1`
	`pause`
	`rem The /a:d switch only lists directories`
	`dir %1 /a:d/o/p`
	`echo.`
	`echo.`
	`echo.`
	`echo  Files in %1`
	`pause`
	`rem The /a:-d switch only lists files`
	`dir %1 /a:-d/o/p`

Exit and save the file.	
Try the batch file for the \WINDOWS directory. The text for the replaceable variable is typed after the name of the batch file.	`sortdir  \WINDOWS`
If you omit the parameter, DOS will use the current directory	`cd \` `sortdir`

Practice 6 - Creating a Batch File to Move Files

Until DOS 6.0, there was no command to **move** files at the system prompt. We had to first copy the files to the target and then delete them from the source.

We are going to write a sophisticated batch file with two replaceable parameters to move files with an extension we specify from the **current** directory to a **target** directory we specify. The batch file will allow us to specify the target at run time.

Task/Comments	Keystrokes
Create a batch file named MOVE.BAT.	`edit MOVE.BAT`
Enter the commands to turn the echo off and clear the screen.	`echo off` `cls`
Using replaceable parameters, enter the command to copy any file (%1) from the current directory to any subdirectory (%2). For the second line, enter a command to delete the file you just copied.	`copy  %1 %2` `del  %1`
Exit and save the file.	`File...Exit, [Enter] to save`
Use EDIT to create a file called TESTFILE in \WP51	`edit \wp51\testfile`
Change to the \WP51 directory	`cd \WP51`
Try the batch file to move TESTFILE from \WP51 to the \TEMP directory.	`move  TESTFILE  \TEMP`
Check the \WP51 and \TEMP directory. The file should be removed from the \WP51 directory and added to \TEMP.	`dir  /on /w` `dir  TEMP`

Practice 7 - Using an IF Statement

Our MOVE batch file is dangerous as it now stands! If you forget to enter the target directory at run time, the file will be copied to the current directory, or if you misspell the target directory name, the copy won't work correctly, but unfortunately the delete will. You will lose your file before you can catch the mistake.

We need to first edit the batch file to test to see if the file was copied. If the file does not exist on the target directory, the batch file will then abort before deleting our file. We'll use the **if not exist** command to test a file's existence and the **goto** command to cause the sequence to jump around the delete command to the end of the batch file. The **goto** jump needs a special line whose name starts with a colon (:).

Task/Comments	Keystrokes
Edit MOVE.BAT.	`edit \MOVE.BAT [Enter]`

Add a test to see if the document does not exist on the directory specified. If the file does not exist in the target directory, the batch file should skip the delete command and continue processing at the line named :error.	Your file should read:
	`echo off`
	`cls`
	`copy %1 %2`
	`if not exist %2\%1 goto :error`
	`del %1`
	`:error`

Run the batch file and specify TESTFILE as the file to move but misspell the target directory name. The file shouldn't be deleted from \WP51 because it wasn't copied to \TEMP.	`move TESTFILE \TMP`

Add a message to the end of the batch file to tell the user that the move did not work properly.	Your file should read:
	`echo off`
	`cls`
	`copy %1 %2`
	`if not exist %2\%1 goto :error`
	`del %1`
	`:error`
	`echo The move did not work properly!`

Run the batch file again. Do not specify a target. Did you get a message telling you that the move didn't work?	`move TESTFILE`
Now try the batch file again and specify TESTFILE as the file to be moved and \TEMP as the target.	`move TESTFILE \TEMP`

Everything should work, except that we get the message that the move didn't work. This happens because the message command comes after the :error line.

Add another **goto** command below the **delete** command to redirect the processing to a line named **:end**. Create a line named **:end** as the last line in your batch file.

Your file should read:

```
echo off
cls
copy  %1  %2
if not exist %2\%1 goto :error
del  %1
goto :end
:error
echo   The move did not work
properly!
:end
```

COPY and Devices

We just saw how we can include the keyboard in a command by calling it CON. We can direct output to the monitor using CON also. Copying **from CON** means the keyboard. Copying **to CON** means the monitor. If we don't stipulate CON, DOS assumes that we mean the monitor or keyboard, as in TYPE WP.BAT [to CON].

The DOS names for the parallel ports to which we hook printers are: LPT1, LPT2, PRN and we can print our file with:

```
C:\>COPY WORDPERF.BAT LPT1
```

The DOS names for the serial ports to which we hook mice and modems are: COM1, COM2.

Cancel

If we want to stop a DOS command or interrupt a batch file that gets into a loop, we use: Ctrl + C. This combination of keys means CANCEL!

For example, if DOS is scrolling a long directory listing, we can interrupt it with Ctrl + C, and when the listing stops, we'll see

```
^C
```

at the point where it stopped.

Screen Print

If we want a hardcopy of whatever's on the screen, like a directory listing, we can use DIR to get the listing on the screen and use the PRINT SCRN (or PrtSc) key to print a copy.

Redirection

Redirection changes the place that a command usually gets information from, or puts it to. Normally, when we type something like:

```
C:\>DIR
```

the output is automatically sent to the screen. But if we wanted the output to go to a file or printer, we could type:

```
C:\>DIR > dirlist.txt
```

```
C:\>DIR > LPT1
```

To add the new information to the end of an existing file, we use a double >>:

```
C:\>DIR >> dirlist.txt
```

Similarly, we can take input from a file rather than from the keyboard:

```
C:\>SORT < dirlist.txt
```

tells DOS SORT command to take its input from the file DIRLIST.TXT and display the results on the screen.

Filters

A **filter** command also redirects information, but also allows us to sort it and select parts of it. There are three filter commands: MORE, FIND, and SORT:

MORE

MORE divides the output of a command into single screenfuls. For example:

```
C:\>MORE < dirlist.txt
```

divides the listing in the file DIRLIST.TXT into single screen displays and shows the first on the screen along with the instruction to press any key for the second screen, and so on.

The MORE command is often used in a **pipe** with another command, as in:

```
C:\>DIR | MORE
```

This pipe generates a directory listing of C:\ and displays it one screen at a time. (The pipe symbol is above the \ key on the keyboard.)

FIND

FIND searches through files for a text string. For example:

```
C:\>FIND "CFU" < handout.txt
```

will search the file HANDOUT.TXT and display occurrences of the text string CFU on the screen. We can redirect the output to a file called EXTRACT.TXT by typing:

```
C:\>FIND "CFU" < handout.txt > extract.txt
```

SORT

SORT sorts the lines of the named file alphabetically. For example, the following command will sort a list of names contained in the file NAMELIST.TXT and display the result on the screen:

```
C:\>SORT < namelist.txt
```

To redirect the sorted file to SORTLIST.TXT:

```
C:\>SORT < namelist.txt > sortlist.txt
```

Pipes

We can use the pipe feature of DOS to combine a number of commands. For example, the following command will list every directory name in the C: drive, select only the directories with DATA as part of the name, and display the names one screen at a time:

```
C:\>DIR C:\ /S /B | FIND "DATA" | MORE
```

Streamlining DOS Commands

DOSKEY

Doskey is a small (3 KB) program included with DOS that maintains a buffer that remembers the previous 20 - 30 DOS commands. To install Doskey, we simply type DOSKEY at the prompt. (Because DOSKEY is so useful, we suggest including the command in your AUTOEXEC.BAT file.)

We access the table using the following keys:

- UP arrow displays the previous command (can be used repeatedly)
- DOWN arrow displays the next command (can be used repeatedly)
- F7 displays the entire table of commands
- F8 cycles through all commands beginning with the letter(s) we type
- F9 prompts us for the number of the command in the table displayed by F7
- PAGE UP displays the oldest command in the list
- PAGE DOWN displays the newest command in the list
- ESC clears the current command from the screen.

For example, F7 may show us a list such as:

```
1: COPY C:\WP51\WPDATA\LETTERS\SMITH.LTR A:
2: DIR C:\DBASE\DBDATA\INVENTRY
3: COPY A:SALESRPT.WP C:\WP51\WPDATA\REPORTS\1991SALE.WP
```

If we press F9, DOS prompts us for a number. Entering 1 would cause the first command to appear on the command line for us to ENTER or modify using the edit keys.

The main difference in edit keys with DOSKEY is that the LEFT ARROW key is non-destructive, that is, it does not work like the BACKSPACE key but merely moves the cursor to the left to allow us to edit the command.

To clear the command list and start afresh, enter Alt + F7.

DOSKEY and Macros

A macro is a named set of commands that are entered once and can be run simply by typing the name. Macros are similar to batch files, but are stored in RAM rather than on a hard disk. They are therefore faster to execute and can be invoked from any directory. Also, macros are limited to 127 characters, and we put all the commands on the same line rather than on separate lines. Unlike batch files, macros are temporary and die when the power does.

For example, to create a macro that automatically displays file names in wide mode:

```
C:\>DOSKEY DW=DIR /W
```

To get the display, simply enter DW at the prompt.

DOS Macros

Macro is short for macro-command -- that is, many commands rolled into one. You create a DOS macro by using the word **doskey** as a command. The name of the macro and the contents of the macro are the parameters for the **doskey** command. The command **doskey ddir=dir/w** creates a macro **DDIR** consisting of the DOS command **dir/w**. To run the macro, type its name at the DOS system prompt (command prompt): **ddir.**

Interestingly, if you typed a space before any command in earlier versions of DOS, the command would not run; in DOS 5.0 and later, it will. That's because of the macro feature. You might create a macro named **dir** containing the command **dir/w**. There is already a DOS command **dir**. What should DOS run when you type the command **dir**? Answer: **dir** with no preceding space will run the macro, and **dir** after a space tells DOS to run the DOS command **dir.**

To change a macro, edit it as you would anything else. If it is stored in a batch file, edit the batch file. If it is stored in buffer memory by Doskey, use Doskey to recall the command from the buffer to the command line, and edit the macro with Doskey editing keys. When you are finished, press **[Enter]** to save the command.

Unlike a batch file, a macro is stored in memory, and is lost when you turn the machine off. To save the macros currently in memory to a batch file on disk, use the **doskey** command with the **/macros** switch, a redirection character, and a batch file name: **doskey/macros filename.bat.** (Put a space before and after the .) You can use Edit to edit this batch file.

To load macros in a batch file into memory, first edit the batch file to add the word **doskey** (and a space) to the beginning of each line containing a macro. Then save and close the file. When you type the batch file's name, the macros will be loaded into memory.

Note: Remember that every macro in the file will have a format like this: macro=command. Try not to mix lines starting with macros and lines starting with other commands in this file.

To delete a macro, issue the command that created the macro, but omit the contents of the macro-command: **doskey ddir=** will delete the macro **ddir** from memory. To delete all macros from memory, press **[Alt+F10]**.

Using DOS Macros

Because each macro is made up of many commands, macros are much like the batch programs you can write for MS-DOS. The biggest difference is that macros are stored only in **memory**, while batch programs are loaded into memory from permanent storage on a disk. You have to recreate your macros for each working session because they disappear from computer RAM every time you reboot or power down.

The way around this limitation is to store your macros as a batch file. Here are some guidelines for macros and how they differ from DOS batch programs:

- Macros are stored in RAM, so they run faster than batch files stored on a disk.
- Macros take memory from your system's command history buffer.
- Macro commands are entered all on one line and are limited to 127 characters.
- Pressing **[Ctrl+C]** halts only one command in a macro; you must use **[Ctrl+C]** to abort each succeeding command. There is no shortcut.
- Special characters used in macros are different from special characters used in batch files.
- You cannot jump from one line to another when in a macro or start other macros from within the current macro.
- You can run a batch program from inside a macro, but you cannot run a macro from inside a batch program.
- In a macro, every command will be displayed as it is executed.

Replaceable Parameters

Macros may include a replaceable parameter ($). The "$1" will be replaced in the file by the second word or character you type at the prompt. The first word will be a command. For example, you might create a macro **doskey findext=dir c:\.$1.** At the command prompt, type **findext bat.** The macro will run the command **dir c:\.BAT** and display all files ending with **.BAT.**

1. Creating Macros with Doskey

1. At the prompt, type **doskey** (space) (the name of the macro)=(the keystrokes) **[Enter]**.
2. To replay the keystrokes, type the name of the macro and press **[Enter]**.

2. Storing Macros on Disk

1. At a DOS system prompt, enter **doskey/macros FILENAME.BAT [Enter]** (for FILENAME substitute the name you want to use).
2. Start Edit and load the batch file that contains your macros.
3. Edit the file to delete any macros you do not want to save.
4. Type **doskey** (and a space) at the beginning of every line that starts with a command.
5. To load all the macros in your batch file into memory, run that batch program from any DOS command prompt.

Practice 1 - Create a DOS Macro to Find .BAT Files

Create a macro to list all the files with a **.BAT** extension in all subdirectories on the current drive:

Task/Comments	Keystrokes
Define a Doskey macro to find all files on the current drive that end in **.BAT.**	`doskey findbat=dir .BAT /s`
Run the macro.	`findbat`

Practice 2 - Create a DOS Macro to Find Files with any extension

Create a macro that allows us to specify the extension at run time.

Task/Comments	Keystrokes
Write a Doskey macro to find files in the current directory and all subdirectories that end in **any** file name extension, pausing between pages.	`doskey findext=dir .$1 /s /p`
Run the macro to find **.EXE** files	`findext EXE`
Run the macro to find .COM files	`findext COM`

Practice 3 - Save DOS Macros on Disk

Macros are only stored in memory, so we need to save them to disk for permanent use. If we save the macros to a batch file, we can run the batch file to install them in memory when we boot up, or automatically run the batch file from our **AUTOEXEC.BAT.**

Task/Comments	`Keystrokes`
Save all current macros to a batch file in the root directory named **MACROS.BAT**	`doskey/macros > MACROS.BAT`
Edit the batch file.	`edit MYMACS.BAT`
Add the command **doskey** and a space at the beginning of each command line.	
Exit Edit and save the file.	

Practice 4 - Using the Saved Macro Batch File

We are going to clear all the entries out of Doskey and run the batch file to load our macros back into Doskey memory.

Task/Comments	`Keystrokes`
Enter the doskey command again with a reinstall switch. This installs a new copy of DOSKEY, deleting any existing macros.	`doskey/reinstall`
Try your **findbat** macro. You should get a message **Bad command or file name.** The macros are no longer in memory.	`findbat`
Run the batch file containing your macros.	`MACROS`
Enter the command to list the macros.	`doskey/macros`
Now try the macro that finds **.BAT** files.	`findbat`
Find all the files with a **.WK1** extension.	`findext WK1`

Section 4: System Optimization

You can use the following methods to speed up your system without taking up additional memory. These methods improve your system's speed by improving the efficiency of your hard disk:

- Delete unnecessary files.
- Use CHKDSK /F to recover lost disk space, and then delete the files it creates.
- Make sure DOS is searching for files in the most efficient order.
- Reorganize files on your hard disk.

1. Deleting Unnecessary Files

Disk space is a valuable system resource. If you need more disk space, an easy solution is to delete unnecessary files. Files types you might want to delete are:

- Program and data files that you no longer use.
- Temporary files that were left on your hard disk when a program ended unexpectedly.
- DOS files that were installed automatically and that you don't use. CAUTION Do not delete any DOS files other than the ones listed below.
- Delete any temporary files created by your programs. Many programs create temporary files while they are running, Some programs store those files in a separate directory specified in your AUTOEXEC.BAT file by using the set command. (Most often, you designate such a directory by using the set command with the TEMP or TMP environment variable.) You should periodically clean out your TEMP directory. (This is not necessary if the TEMP directory is on a RAM disk.) To avoid deleting a temporary file that is currently in use, you should delete files in your TEMP directory only when you are not running any programs.
- If your system is very short on disk space, delete some DOS files. If you plan to delete DOS files, you should first use the DOS Setup program to install DOS on floppy disks. This makes it easy for you to restore individual files later.

DOS files you can delete:

- EMM386.EXE (Memory manager). Delete if you are using a 286 machine, or if you have a 386 or 486 but don't use programs requiring expanded memory
- RAMDRIVE.SYS (RAMDrive memory-disk program, used to speed up your system. Delete if you don't use RAMDrive
- SMARTDRV.SYS (SMARTDrive disk-caching program, used to speed up the system. Delete if you have only conventional memory.
- NLSFUNC.EXE, GRAFTABL.COM, KEYB.COM, *.CPI, COUNTRY.SYS, DISPLAY.SYS, KEYBOARD.SYS, PRINTER.SYS. These files provide international support only.
- EXE2BIN. Delete this file if you don't do programming.

CAUTION: Never delete COMMAND.COM, IO.SYS, MSDOS.SYS, or DBLSPACE.BIN. (IO.SYS, MSDOS.SYS, and DBLSPACE.BIN are hidden files.) If you delete any of them, your system will not start.

2. The CHKDSK Command

You can use CHKDSK to recover lost allocation units that are taking up space on your hard disk. An allocation unit is the smallest unit of a hard disk that can be allocated to a file. Allocation units can get lost when a program ends unexpectedly, leaving temporary files on the hard disk without saving or deleting them properly. Over time, lost allocation units can accumulate and take up disk space.

When you use the /f switch with the **CHKDSK** command, **CHKDSK** converts lost allocation units to visible files that you can examine and delete.

CAUTION: Before using **CHKDSK /f,** make sure you aren't running any programs. You may need to disable memory-resident programs in your CONFIG.SYS and AUTOEXEC.BAT files and restart your system. You might lose data it you use this command while programs are running. Also make sure you quit all programs before using **CHKDSK.** If you use the Fastopen program, the SMARTDrive program, or other memory-resident programs, disable the corresponding commands in your CONFIG.SYS file and restart your system, to ensure that those programs don't interfere with the process.

Use the **CHKDSK /f** command to:

- Ensure that there are no lost allocation units on your disk
- Check your hard disk before compacting your disk (running a defragmentation utility)
- Check your hard disk after a program ends unexpectedly

To clean up lost allocation units by using **CHKDSK**:

1. Quit all programs.
2. Change to the hard disk you want to clean up
3. Type **CHKDSK /F.** The **/F** switch finds and recovers any lost allocation units
4. If **CHKDSK** finds any lost allocation units, it prompts you to convert them to files. If you want to inspect the contents of the lost allocation units before deleting them, type "y" for yes. (If you're sure the lost allocation units don't contain information you want, type "n" for no. The **CHKDSK** command deletes the information, and you can skip the remaining steps in this procedure.) If you answer yes, **CHKDSK** converts any lost file allocation units to visible files with filenames like FILE0001.CHK in your root directory. The **CHKDSK** command also displays information about the disk it just checked.
5. Use the **type** command to examine the CHK files. (You can also use the View File Contents command on the File menu in DOS Shell.) For example, to examine the file FILE0001.CHK, you would type "type **file0001.chk**". Sometimes a .CHK file contains information you want to keep. For example, if a text-editing program ends before you save your edits, you might find your lost edits in a recovered CHK file.
6. Delete any CHK files you don't want.

3. SCANDISK Command

ScanDisk is a disk analysis and repair tool that checks a drive for errors and corrects any problems that it finds. To check the current drive for disk errors, use the following syntax:

```
SCANDISK
```

See DOS Help for details of optional parameters.

Problems fixed by ScanDisk include:

- File allocation table (FAT)
- File system structure (lost clusters, crosslinked files)
- Directory tree structure
- Physical surface of the drive (bad clusters)
- DoubleSpace volume header (MDBPB)
- DoubleSpace volume file structure (MDFAT)
- DoubleSpace compression structure
- DoubleSpace volume signatures
- MS-DOS boot sector

SCANDISK can check hard drives, DoubleSpace drives, floppy disk drives, RAM drives and memory cards.

4. Help DOS Find Files Quickly

When you type a command or start a program, DOS must find the executable file before it can carry out the command or start the program. If you type the full path and filename of the file, DOS can find and carry out the command or run the program almost immediately. If you type only the filename, DOS searches for the program file as follows:

1 . DOS looks for the program file (.EXE or .COM) in the current directory.

2. If the file is not in the current directory, DOS looks for the file in the directories specified by your path= command. It searches the directories in the order they appear in the **path=** command in the AUTOEXEC.BAT. This search can take time, particularly if your path contains many directories or if your directories contain many files. The fewer directories and filenames DOS must search through, the faster the response will be.

If your disk has one or two directories that contain frequently used program files, list those directories first in your **path=** command. For example, suppose all your DOS batch (.BAT) programs are in the directory C:\BATFILES and the programs you use most often are in the directory C:\PROGRAMS. An efficient **path** command would be:

```
path=c:\;c:\batfiles;c:\programs;c:\dos;c:\wp51
```

Keep the number of files in each directory to 150 or less. This reduces the time DOS spends searching.

5. Compacting Your Hard Disk to Improve Speed

Over time, as programs read files from and write files to your hard disk, they can become fragmented. Fragmentation occurs when a file, instead of being stored in contiguous sectors of the disk, is broken into fragments that are stored in different locations on the disk. Although fragmentation doesn't affect the validity of the data, it takes much longer to read them from the disk and to write them back to the disk.

CAUTION: Run the disk-compaction program directly from DOS after quitting all other programs, including memory-resident programs.

The advantages of using a disk-compaction program are:

- Faster read/write. This, in turn, speeds up system performance.
- Significantly decrease in the time it takes for programs to start.
- Easy to implement.

Recommendations

- Compact your hard disk(s) regularly to help keep your system's performance from degrading because of file fragmentation.
- Compact your hard disk immediately before installing new programs on it. (This is less important if you have been compacting the disk regularly.)
- Delete any unnecessary files, and then use CHKDSK /F
- Make sure you quit all other programs before running compaction.

DoubleSpace

DOS 6's DoubleSpace is an integrated disk compression that increases available disk space by compressing files. You can use DoubleSpace to increase available space on both hard disks and floppy disks. Doublespace MUST be launched from the DOS prompt. For more information in DOS, type **help dblspace**.

The first time you run DBLSPACE, it starts the DoubleSpace Setup program. DoubleSpace Setup compresses your hard disk drive and loads DBLSPACE.BIN into memory. DBLSPACE.BIN is the part of MS-DOS that provides access to compressed drives. Setup creates an uncompressed H: drive on your hard disk to hold important system files, and then compresses the remainder of the C: drive.

Thereafter, when you run the DBLSPACE command without specifying any switches or parameters, the DoubleSpace program starts. This program lists your compressed drives and provides menu commands for working with them. You can perform all DoubleSpace tasks either from within the DoubleSpace program or from the DOS command line.

DBLSPACE.BIN and DBLSPACE.SYS

DBLSPACE.BIN is the part of DOS that provides access to your compressed drives. When you start your computer, DOS loads DBLSPACE.BIN along with other operating system functions, before carrying out the commands in your CONFIG.SYS and AUTOEXEC.BAT files. DBLSPACE.BIN always loads in conventional memory, since it loads before device drivers that provide access to upper memory.

The DBLSPACE.SYS device driver does not provide access to compressed drives; it simply determines the final location of DBLSPACE.BIN in memory. When loaded with a DEVICE= command, the DBLSPACE.SYS device driver moves DBLSPACE.BIN from the top to the bottom of conventional memory. When loaded with **DEVICEHIGH=**, DBLSPACE.SYS moves DBLSPACE.BIN from conventional to upper memory, if available as in the following fragment of CONFIG.SYS.

```
DOS=HIGH,umb
FILES = 75
BUFFERS = 60
STACKS=9,256
DEVICEhigh=C:\CPCSCAN.SYS 3e0 2 1
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /e:512 /p
DEVICEHIGH=C:\DOS\DBLSPACE.SYS /MOVE
```

How DoubleSpace assigns drive letters

When you install DoubleSpace, it creates a new drive and assigns a drive letter to that drive. DoubleSpace skips the first four available drive letters and assigns the next available drive letter to the new drive. For example, if your computer has only drives A, B, and C, DoubleSpace skips letters D, E, F, and G, and assigns drive letter H to the new drive.

When assigning letters to additional drives (for example, if you compress another drive), DoubleSpace works backwards from the first drive letter it assigned. In the example above, DoubleSpace would next assign the letter G.

DoubleSpace attempts to avoid drive-letter conflicts with drives created by FDISK, RAMDrive, networks, or other installable device drivers that assign drive letters. However, if a drive-letter conflict does occur, DoubleSpace resolves the conflict by reassigning its drive letters.

DoubleSpace Drives

After you set up DoubleSpace, your computer's drives will be configured differently:

- Drive C will be compressed and will contain more free space than it did before. You will use drive C just as you did before you set up DoubleSpace.
- A new drive will have been created that is not compressed. This drive is used to store files that must remain uncompressed. For example, the Windows permanent swap file does not work properly when it is compressed; if your Windows permanent swap file was previously located on drive C, DoubleSpace moves it to the new uncompressed drive.

DoubleSpace also uses the new uncompressed drive to store important system files such as IO.SYS, MSDOS.SYS, DBLSPACE.BIN, DBLSPACE.INI, and DBLSPACE.000. Most of the files it contains have the Hidden attribute so they won't show if you use dir. Many also have the Read-Only attribute. To view the files, type **dir /a** at the command prompt. DBLSPACE.000 is the **compressed volume file** (CVF) that stands for the compressed C: drive. Notice its file size.

```
H:\>dir /a
 Volume in drive H has no label
 Volume Serial Number is 1A85-7AD3
 Directory of H:\

IO              SYS                40470     03-10-93         6:00a
MSDOS           SYS                38138     03-10-93         6:00a
386SPART        PAR              1036288     05-07-93         9:41a
IMAGE           IDX                   29     04-05-93         3:24p
DBLSPACE        BIN                51214     03-10-93         6:00a
DBLSPACE        INI                   91     04-05-93         6:58p
DBLSPACE        000            208267264     05-07-93         9:40a
      7 file(s)  209433494 bytes
                 2609152 bytes free
```

CAUTION: UNDER NO CIRCUMSTANCES DELETE DBLSPACE.000. Do not tamper with the hidden files on the new drive. If you change or delete these files, you might lose all the files on drive C.

Section 5: Computer Memory

All programs and data must be loaded into memory in order to run or be processed. Most computers come with 640Kbytes of memory, called **conventional memory**, but some programs and data, especially graphics, need more memory. The 640K figure is based on the ability of early versions of DOS to manage memory, although with DOS 5.0, that figure increases to 1Mb. Many 286 and 386 machines come with an additional 384K, making a total of 1Mb. The additional memory is called **upper memory**.

We can increase the amount of memory a computer has beyond conventional memory by adding two further types of memory:

- Extended Memory, or XMS
- Expanded Memory, or EMS

Most programs designed to run under DOS use conventional memory, and for them to run in extended or expanded memory, we must use a memory manager program because programs designed for conventional memory can only address memory locations up to 640K. Such programs need special support to address memory above that.

Extended Memory

We can add **extended memory** to computers that have the 80286 microprocessor or higher in the form of chips or groups of chips, often called Single-Inline Memory Modules, or SIMMs. These come in units of memory, such as 256K or 1Mb. Extended memory is anything over the 640Kb of conventional memory, 640K plus the additional 384K on 286 and 386 machines.

We must also install an extended memory manager, such as the HIMEM.SYS that comes with DOS 5.0. This program organizes the various programs that want to use extended memory and prevents accidental overlays of one program by another. (HIMEM.SYS conforms to the industry standard, eXtended Memory Specification, or XMS. Programs requiring extended memory also conform to this standard.)

DOS can run in extended memory, leaving conventional memory for application programs, and Windows makes particularly effective use of extended memory. In standard and enhanced modes, Windows fully uses all available extended memory when running applications designed to take advantage of it. However, many older non-Windows applications cannot work with extended memory, even when run from Windows.

HIMEM.SYS

Using HIMEM.SYS:

- Makes extended memory available to programs that use extended memory according to XMS (the Extended Memory Specification).
- Prevents system errors that can result when programs make conflicting memory requests.
- In conjunction with EMM386, enables DOS to run in extended memory to conserve conventional memory.
- In 386 and 486 machines, enables use of the upper memory area to conserve conventional memory. (To do this, you also need to install EMM386.)
- Enables EMM386 to use extended memory to emulate expanded memory for programs that need expanded memory.

HIMEM uses a small amount of conventional memory, and might not be compatible with older programs that allocate extended memory directly, rather than by using an extended-memory manager.

Recommendations for using HIMEM:

- Install HIMEM if you have an 80286, 80386, or 80486 system with extended memory.
- Make sure the device command for HIMEM appears in your CONFIG.SYS file before any commands that start device drivers or programs which use extended memory. For example, since EMM386 uses extended memory, the device command for EMM386 must appear after that for HIMEM.

To install HIMEM:

1. Use a text editor such as DOS Editor to open CONFIG.SYS.
2. Add a device command for HIMEM.SYS to the beginning of CONFIG.SYS. This device command must come before any device commands for device drivers that use extended memory. The command line for HIMEM specifies the location of the HIMEM program file, how HIMEM should manage memory, and your system type. The following command runs HIMEM, using default values:

```
device=c:\dos\himem.sys
```

3. Save the changes to your CONFIG.SYS file.
4. Reboot with CTRL+ALT+DEL.

Expanded Memory

We can add **expanded memory** to a computer by means of a board that plugs into an expansion slot. The additional memory is partitioned into 16Kb pages. The proprietary board comes with its own expanded memory manager, and whenever an application program wants to access an area of expanded memory, the manager copies data to or from the pages into an upper memory area (see below) where it can be accessed by the program. Because the manager works with 16K pages, access is usually slower than with extended memory. Also, the application program must have been designed to work with the manager.

With expanded memory boards such as AST's RAMpage or Intel's Above Board, we can define some or all the expanded memory to be extended memory. Windows works best if all the memory is defined as extended.

Expanded memory and the manager must conform to the industry standard, Expanded Memory Specification, or EMS.

If a program was written to use expanded memory and we only have extended memory, we can use the device driver EMM386 that uses extended memory to simulate expanded memory. (Only 386 machines and later.)

Windows and Windows applications do not use expanded memory, but if we are running a non-Windows application under Windows in 386 enhanced mode that uses expanded memory, we must configure the expanded memory board as extended memory, and then use EMM386 to simulate expanded memory from the extended memory.

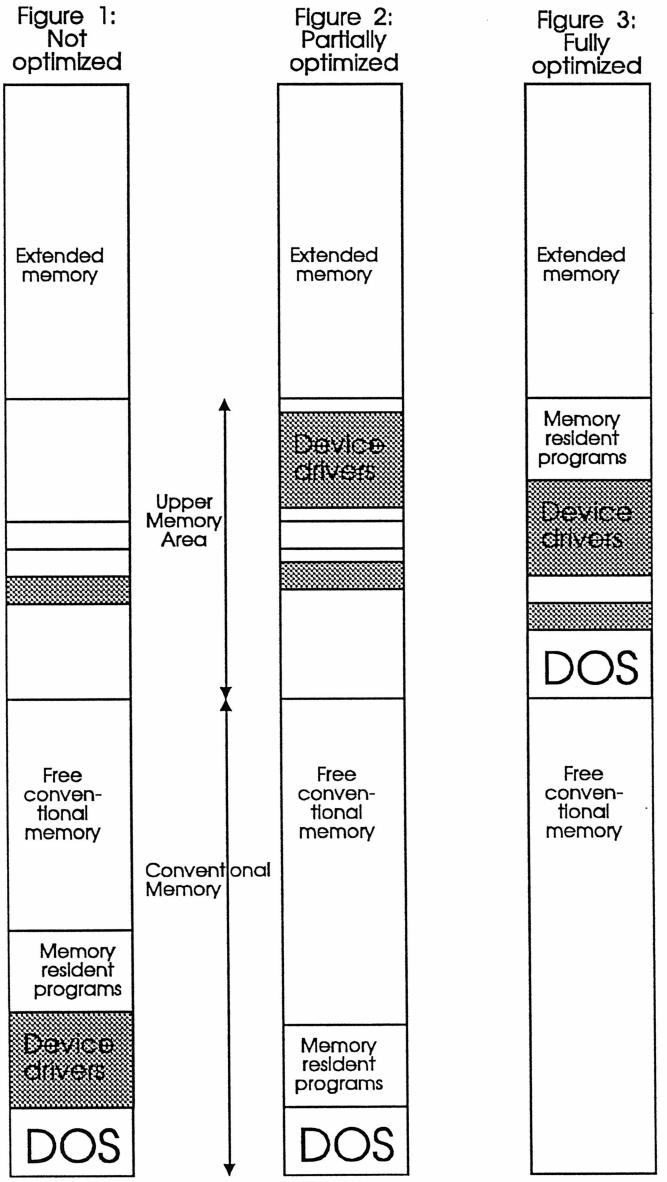
Upper Memory Area

Most systems today have an additional 384K of memory called the Upper Memory Area. It is reserved for system programs like monitor and disk drivers, and expanded memory managers. Application programs do not use this area, and it is not considered part of the computer's total memory. EMM386 is required to manage the UMA and make it available as extended memory.

Using the Upper Memory Area

On 386 machines and later, we can save conventional memory by running device drivers in upper memory blocks, or UMBs. In Figure 1, much of the conventional memory is taken up by DOS, drivers, and memory-resident programs, with most upper memory left empty, and therefore unused.

In Figure 2, we see some drivers moved into upper memory, freeing up conventional memory, and in Figure 3, we see all drivers and most of DOS running in upper memory, with much more conventional memory freed up for application programs.



Optimizing Memory

Insufficient memory available to a program prevents the program from running, but a program may not run even if it looks as if there's sufficient memory. In this section, we look at freeing up conventional memory and making full use of expanded and extended memory.

Freeing Conventional Memory

All programs require conventional memory in order to run. If a program fails to run because of insufficient memory, the problem is most often because of a shortage of conventional memory.

You can make more conventional memory available to your programs by minimizing how much memory DOS, installable device drivers, and other memory-resident programs use. A program can use only the conventional memory that is available when you start it. If memory-resident programs are already using memory, the program cannot use that memory.

To free conventional memory for use by programs:

- Run DOS in extended memory instead of in conventional memory, if your system has extended memory
- Streamline your CONFIG.SYS and AUTOEXEC.BAT files so that they don't start unnecessary memory-resident programs.
- Run device drivers and other memory-resident programs in the upper memory area instead of in conventional memory if you have an 80386 or 80486 computer.

MEMMAKER

DOS 6 provides a utility program called **MEMMAKER** that analyzes how your system is using memory and makes recommendations for improvement. If you accept its recommendations, MEMMAKER changes your AUTOEXEC.BAT and CONFIG.SYS, and reboots your system to test the changes. To use MemMaker, you must have a 386 or 486 and extended memory. **Do not use this command when you are running Windows.**

MemMaker modifies your CONFIG.SYS and AUTOEXEC.BAT files so that your device drivers and other memory-resident programs use less conventional memory. MemMaker frees conventional memory by loading some of those device drivers and programs into the upper memory area.

The upper memory area is the range of memory addresses between 640KB and 1024KB that is normally set aside for use by hardware expansion cards and drivers. However, on most computers, hardware expansion cards do not use up the entire upper memory area. The unused areas of upper memory are called upper memory blocks (UMBs). You can use UMBs for running installable device drivers and other memory-resident programs. Moving these programs out of conventional memory makes more conventional memory available for other programs.

Note The EMM386 memory manager makes UMBs available by mapping extended memory to unused addresses in the upper memory area. Because of this, running programs in upper memory uses up some extended memory. Therefore, you might not need or want to run MemMaker if you use only Windows and Windows-based applications. Windows and Windows-based applications need as much free extended memory as possible.

Running MemMaker frees conventional memory, but can result in less free extended memory. If you run DOS-based applications under Windows, run MemMaker. DOS-based applications need free conventional memory even when running under Windows.)

Running MemMaker

1. MemMaker asks you if you use any programs that require Expanded Memory. If you do, MemMaker asks you more about them.

2. MemMaker then restarts your computer, and as each device driver and memory-resident program starts, MemMaker determines the program's memory requirements by monitoring how it allocates memory.

3. MemMaker restarts your computer again and uses the information it gathered to determine the optimum memory configuration for your computer, fitting your device drivers and memory-resident programs into the available UMBs as efficiently as possible. To do this, MemMaker considers thousands of possible memory configurations before selecting the most efficient one. Even so, this process usually takes only a few seconds.

4. When its calculations are complete, MemMaker makes the necessary changes to your CONFIG.SYS and AUTOEXEC.BAT files, and then displays a screen stating that it is ready to restart your computer using the new configuration.

5. To restart your computer with its new configuration, press ENTER. Your computer starts with its new memory configuration. Watch carefully as device drivers and programs display their startup messages, and note any unusual messages.

6. After your computer has restarted, MemMaker displays a screen that prompts you to specify whether your system appears to be working properly.

Enhanced MEM command

An enhanced **mem** command provides more information about the memory your computer is using and the programs that are loaded in memory. For more information, type **help mem** .

Displays the amount of used and free memory on your computer. You can use the MEM command to display information about allocated memory areas, free memory areas, and programs that are currently loaded into memory.

Syntax: MEM [/CLASSIFY | /DEBUG | /FREE | /MODULE modulename] [/PAGE]

Switches

/CLASSIFY: Lists the programs that are currently loaded into memory and shows how much conventional and upper memory each program is using. MEM /CLASSIFY also summarizes overall memory use and lists the largest free memory blocks. You can use the /CLASSIFY switch with /PAGE but not with other MEM switches. You can abbreviate /CLASSIFY as /C.

/DEBUG: Lists the programs and internal drivers that are currently loaded into memory. MEM /DEBUG shows each module's size, segment address, and module type, summarizes overall memory use, and displays other information useful for programming. You can use the /DEBUG switch with /PAGE but not with other MEM switches. You can abbreviate /DEBUG as /D.

/FREE: Lists the free areas of conventional and upper memory. MEM /FREE shows the segment address and size of each free area of conventional memory, and shows the largest free upper memory block in each region of upper memory. MEM /FREE also summarizes overall memory use. You can use the /FREE switch with /PAGE but not with other MEM switches. You can abbreviate /FREE as /F.

/MODULE programname: Shows how a program module is currently using memory. You must specify the program name after the /MODULE switch. MEM /MODULE lists the areas of memory the specified program module has allocated and shows the address and size of each area. You can use the /MODULE switch with /PAGE, but not with other MEM switches. You can abbreviate /MODULE as /M.

/PAGE: Pauses after each screen of output. This switch can be used with any of the other MEM switches.

```
Modules using memory below 1 MB:
Name           Total      =   Conventional   +   Upper Memory
MSDOS        50141   (49K)      50141   (49K)         0    (0K)
SETVER         784    (1K)        784    (1K)         0    (0K)
NAV&          8080    (8K)       8080    (8K)         0    (0K)
HIMEM         1168    (1K)       1168    (1K)         0    (0K)
EMM386        3120    (3K)       3120    (3K)         0    (0K)
COMMAND       3168    (3K)       3168    (3K)         0    (0K)
win386       35520   (35K)       3040    (3K)     32480   (32K)
MODE           480    (0K)        480   (0K)          0   (0K)
SHARE         6208    (6K)       6208    (6K)         0    (0K)
WIN           1696    (2K)       1696    (2K)         0   (0K)
COMMAND       3200    (3K)       3200    (3K)         0    (0K)
CPCSCAN       9664    (9K)          0    (0K)      9664    (9K)
DBLSPACE     44496   (43K)          0    (0K)     44496   (43K)
SMARTDRV     28816   (28K)          0    (0K)     28816   (28K)
VSAFE        22896   (22K)          0    (0K)     22896   (22K)
MOUSE        16352   (16K)          0    (0K)     16352   (16K)
DOSKEY        4144    (4K)          0    (0K)      4144    (4K)
Free        574032  (561K)     574032  (561K)         0    (0K)

Memory Summary:
Type of Memory    Total       =      Used      +      Free
Conventional   655360 (640K) =81328 (79K) +574032 (561K)
Upper          158848 (155K) = 158848 (155K) +0   (0K)
Adapter RAM 393216  (384K) =393216 (384K) +0  (0K)
Ext (XMS)  15569792 (15205K) =14521216 (14181K) +1048576  (1024K)

Total    16777216  (16384K) =15154608 (14799K) +1622608 (1585K)
Total < 1 MB  814208  (795K) =240176 235K) +574032  (561K)
Largest executable program size         574016   (561K)
Largest free upper memory block              0     (0K)
MS-DOS is resident in the high memory area.
```

MEM Notes:

1. Specifying the /PAGE switch automatically

You can use the DOSKEY program to automatically add the /PAGE switch to the MEM command. Then, each time you use MEM, it will pause after each screenful of information even if you don't type the /P switch on the MEM command line. To do this, add the following commands to your AUTOEXEC.BAT file:

c:\dos\doskey
doskey mem=mem.exe $* /p

2. Displaying memory status

DOS displays the status of extended memory only if you have installed memory above the 1-megabyte (MB) boundary in your system. MS-DOS displays the status of expanded memory only if you use expanded memory that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS). MS-DOS displays the status of the upper memory area only if a UMB provider such as EMM386 is installed and the command DOS=UMB is included in the CONFIG.SYS file.

Enhanced LOADHIGH and DEVICEHIGH

Enhanced loadhigh and devicehigh commands enable you to specify the memory region in which you want to load a program. For more information, type **help loadhigh** or **help devicehigh**.

To use the LOADHIGH and DEVICEHIGH commands, you must include the **DOS=UMB** command in your CONFIG.SYS file and an upper-memory-area manager must be installed such as EMM386.EXE. To install EMM386, add a DEVICE command to your CONFIG.SYS file **after** the DEVICE command for HIMEM.SYS extended-memory manager. If you do not specify DOS=UMB, all device drivers are loaded into conventional memory, as if you had used the DEVICE command.

How LOADHIGH works
When you use the LOADHIGH command to load a program, DOS tries to load it into the upper memory area. If there is insufficient space in upper memory, DOS loads the program into conventional memory. To determine which UMBs the program is using, use MEM /M command and specify the program name as an argument.

Using LOADHIGH in AUTOEXEC.BAT
Include LOADHIGH= in AUTOEXEC.BAT. (MemMaker automatically adds any necessary LOADHIGH commands to the AUTOEXEC.BAT.)

DEVICEHIGH Notes

Installing HIMEM.SYS and a UMB provider
DEVICE=c:\dos\himem.sys installs HIMEM.SYS and **DEVICE=c:\dos\emm386.exe** installs the Upper Memory Block (UMB) provider. These commands must appear before any DEVICEHIGH commands in your CONFIG.SYS file. If no upper memory area is available or if there is not enough upper memory area available to load the device driver you specified with the DEVICEHIGH command, DOS will load it into conventional memory (as if you had used the DEVICE command).

Running DOS in Extended Memory

Normally, DOS runs in conventional memory. This makes less conventional memory available to programs. However, if your system has extended memory, DOS can run in extended memory. When it does, it uses the first 64K of extended memory, called the high memory area (HMA). Because few programs use the HMA, it makes sense to run DOS there. (If your system has extended memory, the DOS Setup program normally installs DOS so that it will automatically run in HMA.)

The advantages of running DOS in extended memory are:

- Frees conventional memory.
- Works on any computer that has extended memory.
- Uses HMA, a part of extended memory that few programs use.
- Loads most of HIMEM into the HMA, freeing even more conventional memory.

To load DOS into HMA:

1. Use a text editor such as DOS Editor to open CONFIG.SYS.
2. Make sure that CONFIG.SYS contains the following commands:

```
device=himem.sys
dos=high
```

These commands first load the HIMEM extended-memory manager and then load DOS into extended memory.

3. Save the changes to your CONFIG.SYS file.
4. Reboot with CTRL+ALT+DEL.

Streamlining Start up Files

When you start your system, the commands in your CONFIG.SYS and AUTOEXEC.BAT files can start device drivers and other programs that use memory. You can make more memory available to programs by removing unnecessary commands from these files. However, to effectively streamline CONFIG.SYS and AUTOEXEC.BAT, you should know the purpose of each of the commands in those files. Use care when changing your CONFIG.SYS and AUTOEXEC.BAT files. If you incorrectly change some values or disable some commands, your system may not function properly.

To streamline CONFIG.SYS:

- Include only essential device drivers
- If your system has extended memory, include a device command for HIMEM.SYS, plus **dos=high**. This saves conventional memory by running DOS in extended memory.
- If CONFIG.SYS contains a device command for SMARTDrive, RAMDrive, or the Fastopen program, disable that command to conserve conventional memory. (SMARTDrive, in particular, can use a lot of conventional memory.) If you use RAMDrive, make sure your RAM disk is in expanded or extended memory, not conventional memory.
- If CONFIG.SYS contains a **buffers=** command, reduce the number of buffers. (Each buffer takes up about 500 bytes.) Because some programs might not run properly if you reduce the number too much, do not specify fewer than 10 or 15 buffers unless using another caching scheme such as SMARTDrive.
- Add a stacks command to limit the number and size of interrupt stacks that DOS uses. By default, DOS uses 0 interrupt stacks for IBM PC, IBM PC/XT, IBM PC-Portable, and compatible machines; and 9 stacks for IBM PC/AT, IBM PS/2, and compatible machines. You can conserve memory by setting stacks to zero with **stacks=0,0**. (If the system occasionally locks up while running Windows 3.0 in 386 enhanced mode, disabling the stacks command might solve the problem.)
- If CONFIG.SYS includes the **lastdrive=** command, set lastdrive to a letter such as J or K rather than Z. (Each letter uses about 100 bytes more than the preceding letter.) If you use a network, this might limit the number of network drives you can use simultaneously.
- If your CONFIG.SYS file contains an **fcbs=** command, set fcbs to 1.

The order of the **device=** and **devicehigh=** commands in CONFIG.SYS affects both the efficient use of memory and the proper operation of the various programs that CONFIG.SYS starts. Start device drivers from your CONFIG.SYS file in the following order:

1. HIMEM.SYS
2. Expanded-memory manager, if your system has physical expanded memory.
3. Any device drivers that use extended memory.
4. EMM386.EXE to manage Upper Memory Area.
5. Any device drivers that use expanded memory.
6. Any device drivers that use the Upper Memory Area.

To streamline AUTOEXEC.BAT:

- Disable commands that start memory-resident programs you don't need.
- If you use a mouse only with Microsoft Windows (which has a built-in device driver), disable any commands that start and enable mouse device drivers such as MOUSE.COM.

Modifying CONFIG.SYS and AUTOEXEC.BAT

Before you change CONFIG.SYS or AUTOEXEC.BAT, back up the existing version of the files. Then, if your changes cause any problems, you can easily restart your computer by using the backup versions, and then correct the modified files. Modify the files as follows:

- Use the SYS command to create a boot disk and copy CONFIG.SYS or AUTOEXEC.BAT to it
- Use a text editor such as DOS Editor to open and edit CONFIG.SYS or AUTOEXEC.BAT
- Disable any commands that load unnecessary device drivers and utility programs. (It's better to disable a command than to delete it, because if you accidentally disable a command you really need, you can restore it easily. To disable a command, insert **rem** at the beginning of the command line.
- Save the file.
- Reboot with CTRL+ALT+DEL.

If the system doesn't start properly, reboot from the floppy disk you created. If you know which command(s) are causing the problem, edit the appropriate file on your hard disk and reboot. Or, to start over, copy the backup version of the files from the floppy disk to your hard disk.